Dynamical Adaptation in ORNs

Srinivas Gorur-Shandilya

February 7, 2014

Dynamical Adaptation in ORNs

Do ORNs exhibit fast adaptation to a flickering stimulus? Can a simple dynamical adaptation model predict ORN responses to flickering inputs? Here, I take data from Carlotta's random stimulus experiments and first check how well a simple linear prediction from the stimulus compares to the data. Then, I study how the instantaneous gain of the actual ORN response w.r.t to the predicted ORN response varies with time, and try to find a objective filter from the stimulus to this instantaneous gain to correct for systematic errors in the linear prediction.

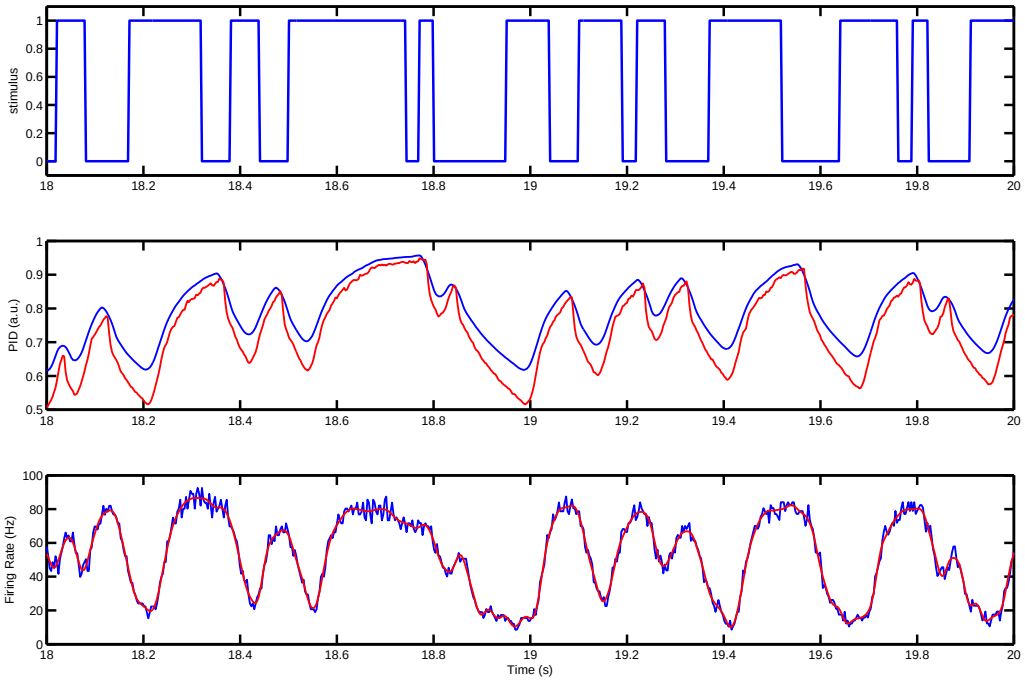
Contents

Data Visualisation

Data from this file will be used for this analysis.

`/data/random-stim/final_2011_06_14_ab3A_1o3ol3X-3_20ml_30sec_30ms_rand.mat`

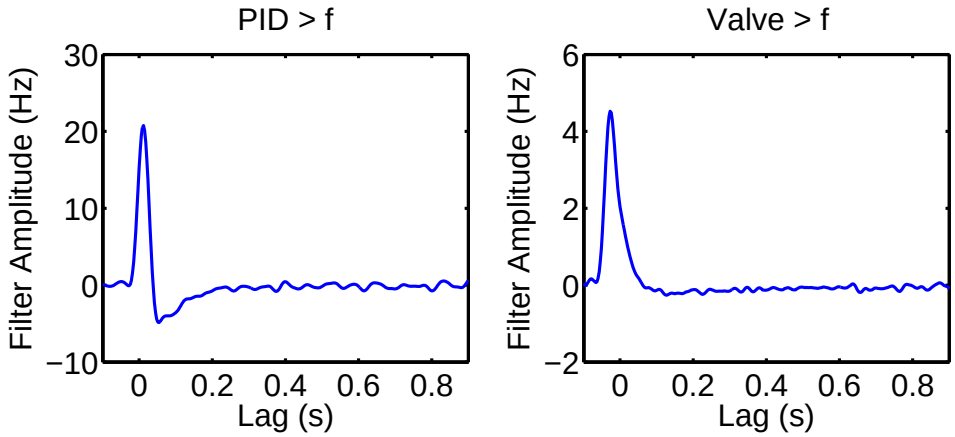
Here, data from simultaneous measurements of the stimulus and from the ORN is shown. The top row shows the valve command signal (a binary signal, high means valve is open). The middle row shows the simultaneous measurement of the odour stimulus with a PID, and the bottom row shows the instantaneous firing rate of the ORN recorded from. In blue is the data computed using Carlotta's code from the raw data. In red is the data computed using my code. Notably different is the firing rate curve, which is much smoother and loses all the high-frequency wiggles. This is because I use a Gaussian smoothing window to do this, while Carlotta uses a boxcar window.



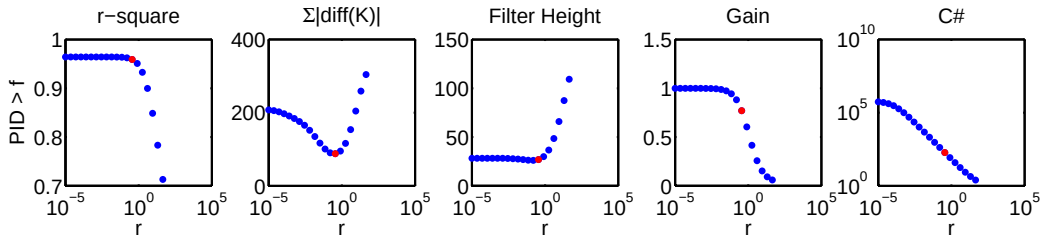
The odour used, the neuron recorded from and the correlation time of the flickering stimulus are in the file name displayed above the plot.

Filter Extraction

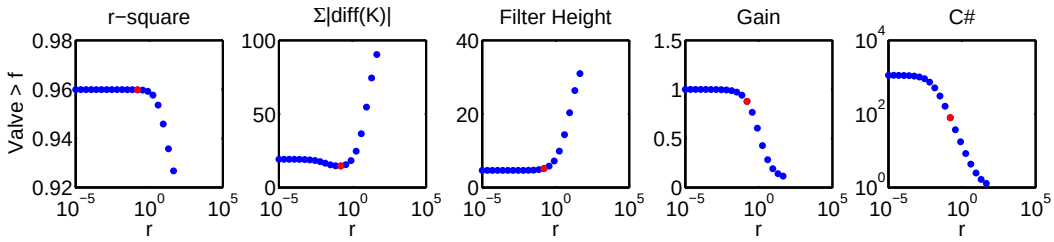
Details about the filter extraction, regularisation methods, and validation with synthetic and real data is listed in (FilterExtraction.pdf). Here, we calculate the best filter (scaled to ensure that gain = 1) using techniques described in that document from the PID (left) and from the Valve (right).



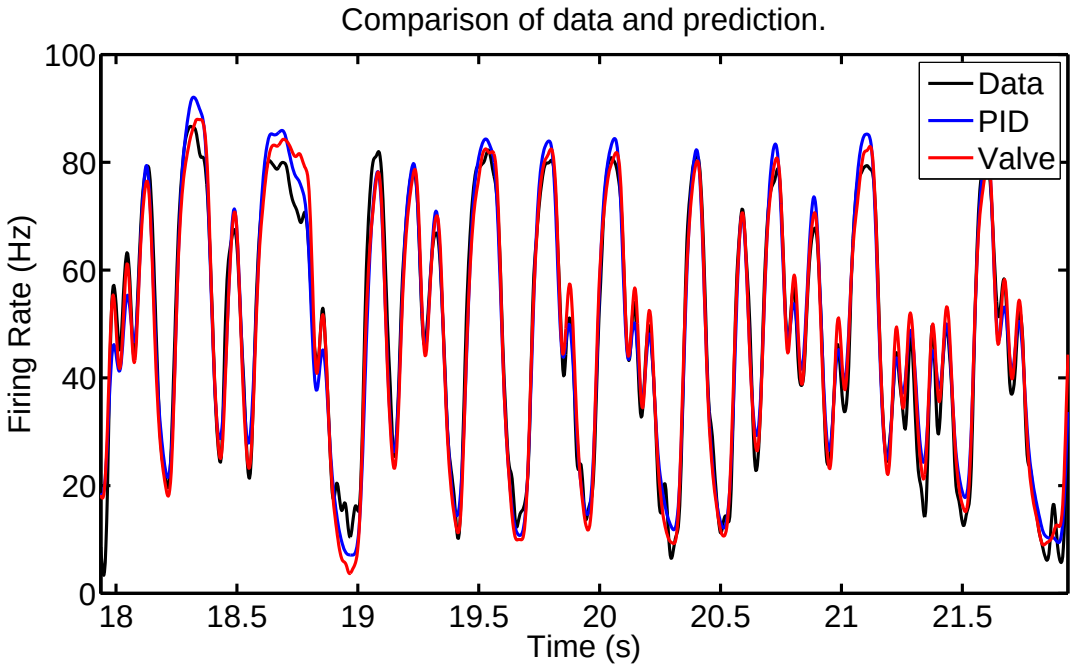
The variation of filter shape with regularisation parameter for the PID > f filter is shown below:



The variation of filter shape with regularisation parameter for the Valve > f filter is shown below:



The convolution of this filter with the input (PID) gives a prediction of the output. In the figure below, the data is shown in black, and the prediction from the PID filter is shown in blue and the prediction from the Valve filter is shown in red. Note that sometimes the prediction can be below zero, which doesn't mean anything physical (firing rate cannot be < 0).



Analysis of Linear Prediction - Which input predicts output better?

The measurement of the stimulus should help us predict the response of the neuron better than just the valve control signal, as we now capture the fast fluctuations of the odour stimulus as it reaches the ORN. Thus, a prediction of ORN firing response from the PID signal should be better than a prediction of the ORN firing response from the valve signal.

The r-square (coefficient of determination) of the PID>f prediction is:

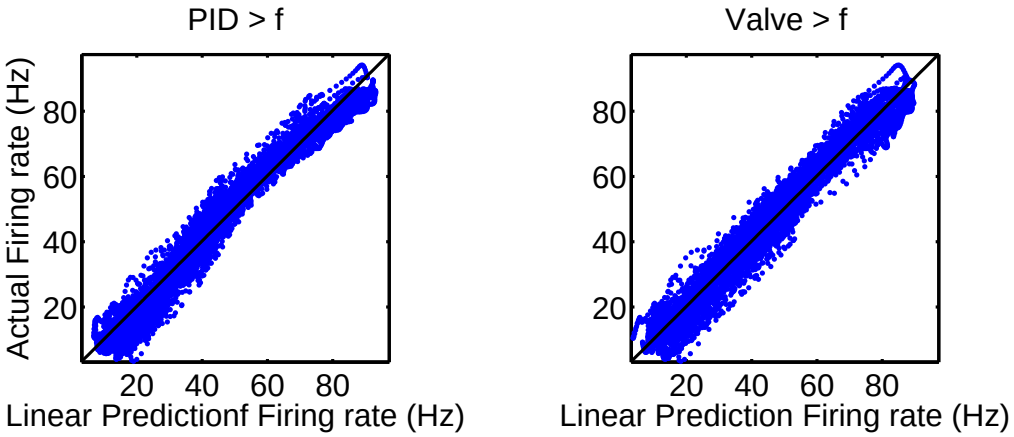
```
0.9592
```

The r-square (coefficient of determination) of the Valve>f prediction is:

```
0.9599
```

Analysis of Linear Prediction - Linearity of Prediction

For a linear filter calculated from the data, a plot of the actual response to the predicted response can be fit with a line of slope 1. Here the actual firing rate is plotted on the X axis and the firing rate predicted from the linear filter is plotted on the Y axis.



The lines are best fits to all the data. The slopes of these lines should be exactly 1 as the "best filters" are supposed to be unit gain. The slope, in fact is:

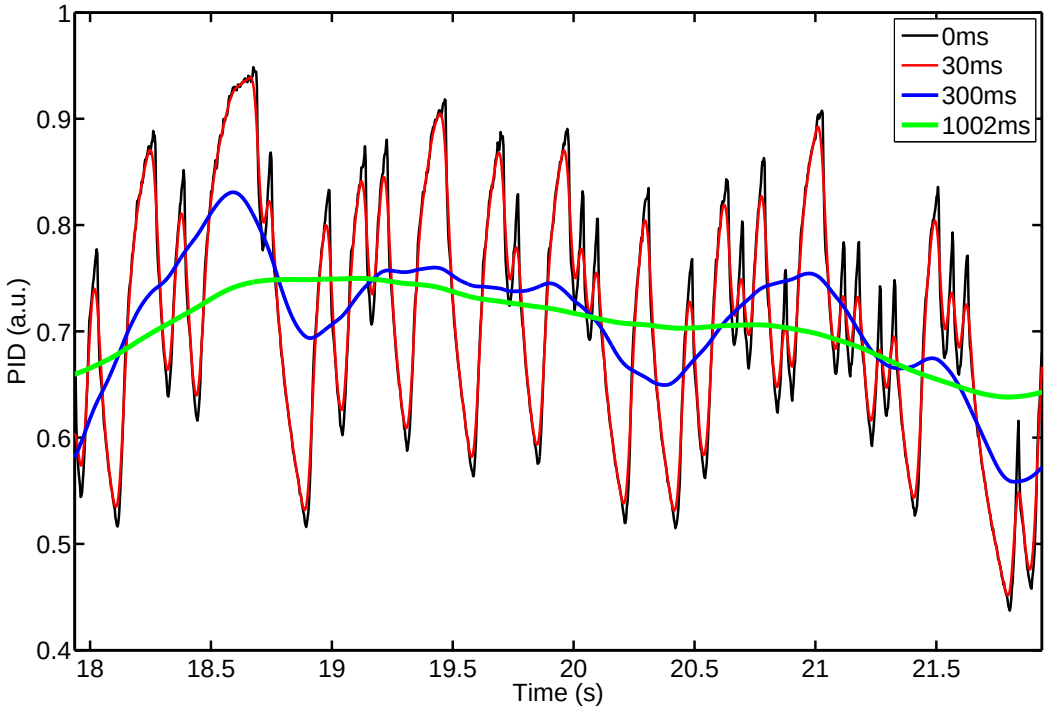
```
1.0000
```

Analysis of Linear Prediction - Response to High and Low Stimuli

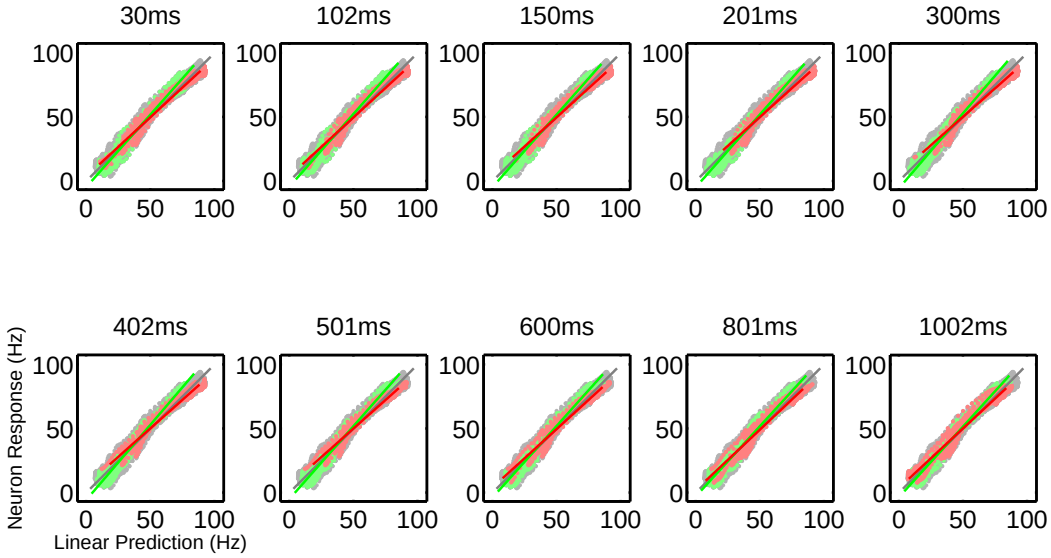
How does the response of the ORN differ for high and low stimuli? Specifically, does the neuron display the characteristics of fast adaptation to this flickering stimulus?

To look at this, we calculate the mean average stimulus over some window history length for every time point t, for various different window history lengths. The effect of this operation is shown in the figure below, where the legend refers to the history window in ms.

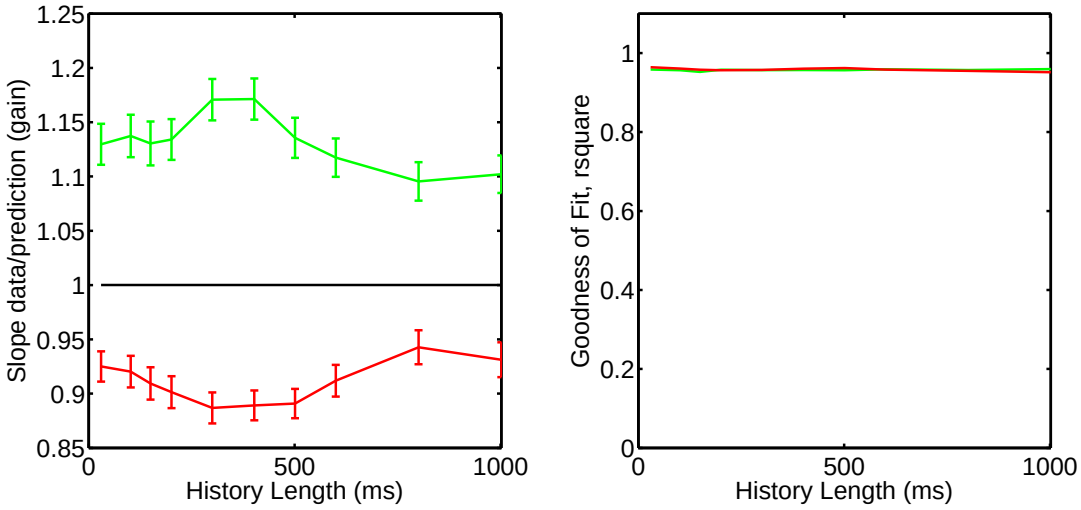
To calcualte the mean average stimulus, I use the *filtfilt* function, which is a zero-phase digital filter that works by passing it through once in the forward direction, and once again in the reverse direction. I previously used the simple *filter* function, but the results are substantially different: now, the separation of gains extends for larger history window lengths.



Now, we separate neuron responses and linear prediction at times when the mean average stimulus is in the lowest 10% or the highest 10%. These points are marked either green (lowest 10%) or red (or highest 10%) in the figures below, while all the data is plotted in grey. Lines are fit to each of these clouds of points, and the slopes (representing the instantaneous gain) is calculated from these lines. The y axis is the actual firing rate, while the x axis is the predicted firing rate. In this analysis the prediction from the PID is used.

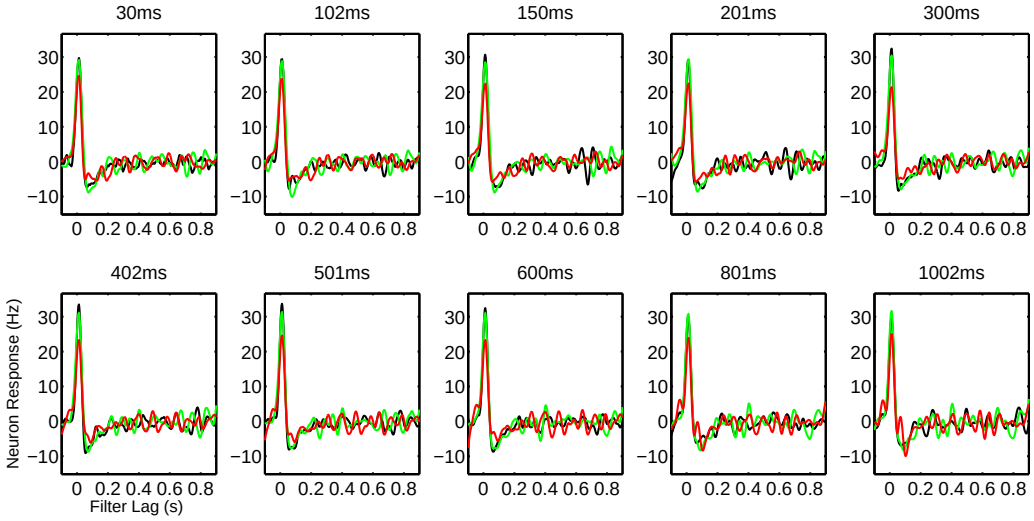


The following plot shows how the slope of the lines of best fit, or the instantaneous gains, varies with the history length. The plot on the right shows the goodness of fit for each fit, indicating the regions where the fit is meaningful.

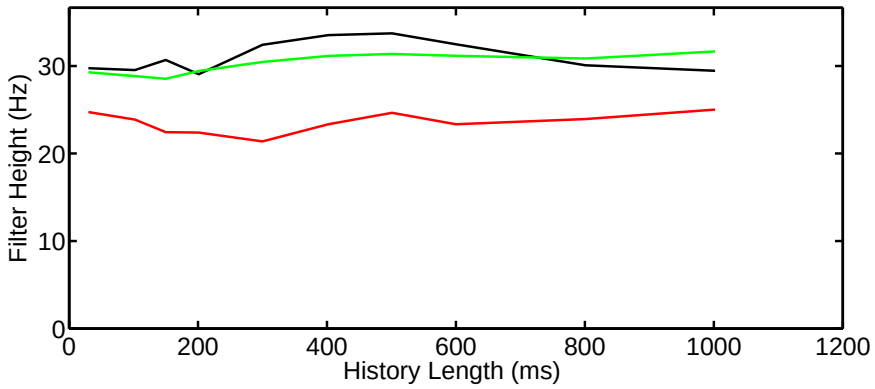


Analysis of Linear Prediction - Filter Variation

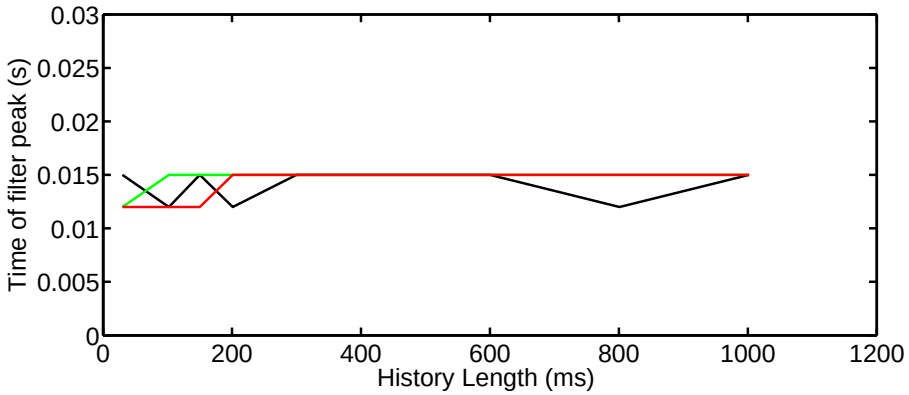
We have segmented the entire data set based on when the stimulus, averaged in some way over the past, into high and low. The following plots show how the filter shape for low, high and middle 10% of the stimuli changes with the history window size. In these plots, only 10% of the data, either the lowest 10% (green), the highest 10% (red) of the middle 10% (black), based on the window smoothing is used to compute each of the filters.



We can also plot the height of the filter *vs.* the history window size:

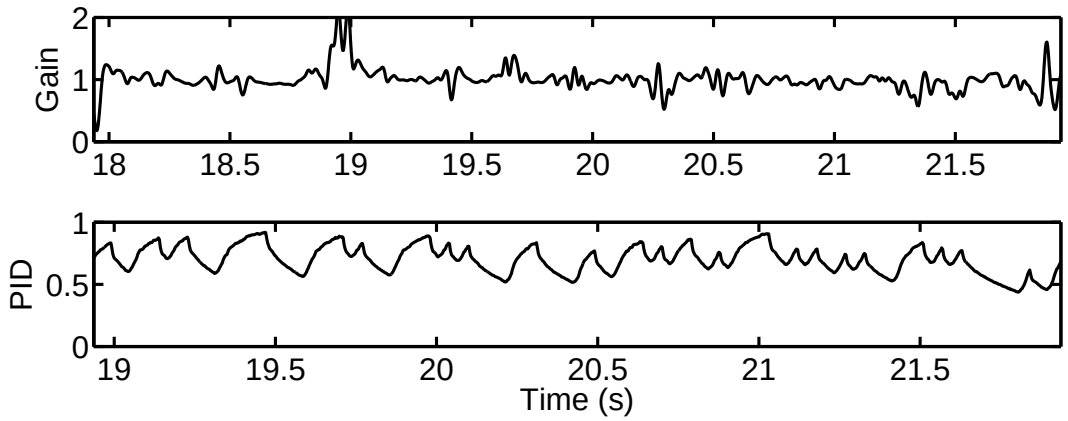


We also expect the low stimulus (high gain) filter to be slower than the high stimulus (low gain) filter. Is this so in the data? Here, I plot the time of peak of each filter *vs.* the history window size.

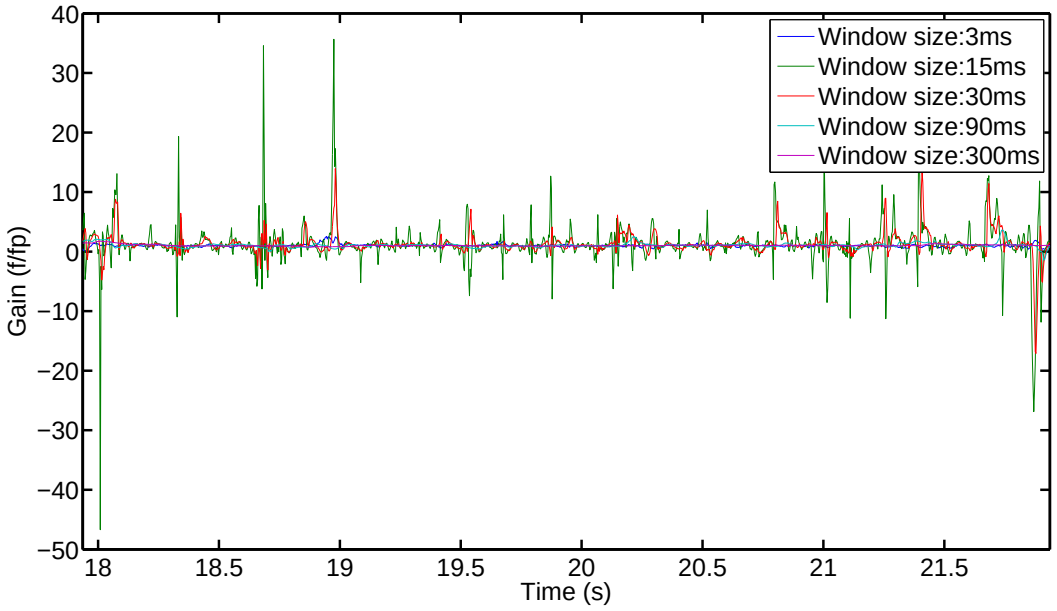


Analysis of Gain: Instantaneous Gain and Fitted Gain

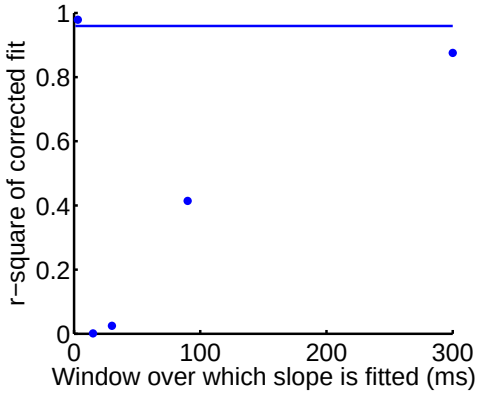
There is a mismatch between the linear prediction and the actual firing rate. Moreover, the instantaneous gain seems to be modulated by something that depends on the past history of the stimulus (see analysis in the previous section). We also know that we cannot predict with a simple linear filter any linear errors that the filter makes. However, there is the possibility of fitting a linear filter to the gain, which is a non-linear function of the filter output. But how do we compute this gain? Here, in the figure below, the instantaneous gain, i.e., the ratio of the actual firing rate to the predicted firing rate, is plotted along with the stimulus.



We can also calculate the moving gain by fitting lines to the data and the prediction collected in bins of size w. The following plots show the gain computed in this way for a few different w. We want to do this because the raw instantaneous gain is very noisy, and predicting it from PID is hard.



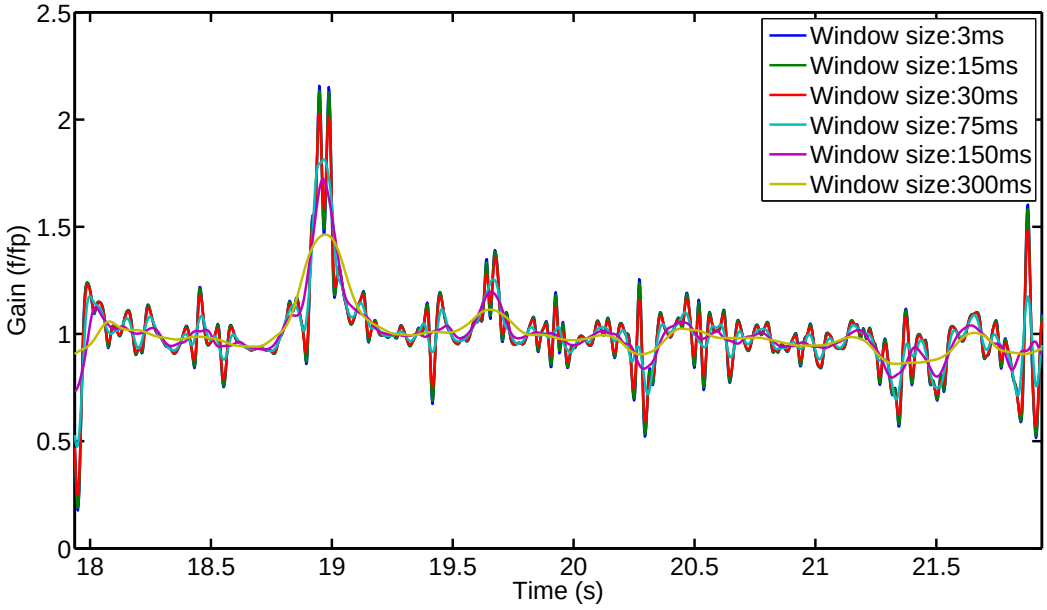
Of these different gain vectors, which fixes the prediction the best? The following plot shows the r-square values of corrected linear prediction, corrected by the gain computed in different ways (lines fit to windows of different lengths). On the y-axis is the r-square, and the x-axis is the window over which the lines are fit. The first point has a r-square of 1, indicating a perfect fit (this is by definition, since the first point is the instantaneous gain). Note that none of the other points exceed the line, which indicates the r-square of the simple linear prediction.



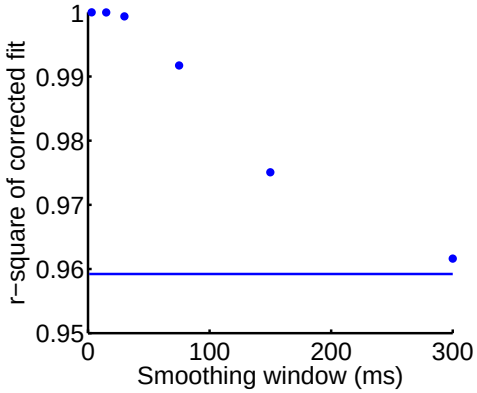
We conclude that this method of finding the gain is not helpful in improving the prediction.

Analysis of Gain: Smoothed gain.

Instead of calculating the instantaneous point-by-point gain, we can also smooth the instantaneous gain over some small window size w. The following plot shows the effect of smoothing for a few window sizes w.



How does smoothing the gain affect our ability to correct the linear prediction? The following plot shows how the r-square of the corrected linear prediction varies with smoothing the gain. Also shown is the line which indicates the linear prediction of the uncorrected simple linear prediction.



From this analysis of the gain, it is clear that if we do want to improve the linear prediction, the best way to do it (apart from the instantaneous gain, which would lead to a perfect prediction), is smoothing the gain in some way. In the next section, we will try to estimate the smoothed gain.

Gain Analysis - Estimating the gain

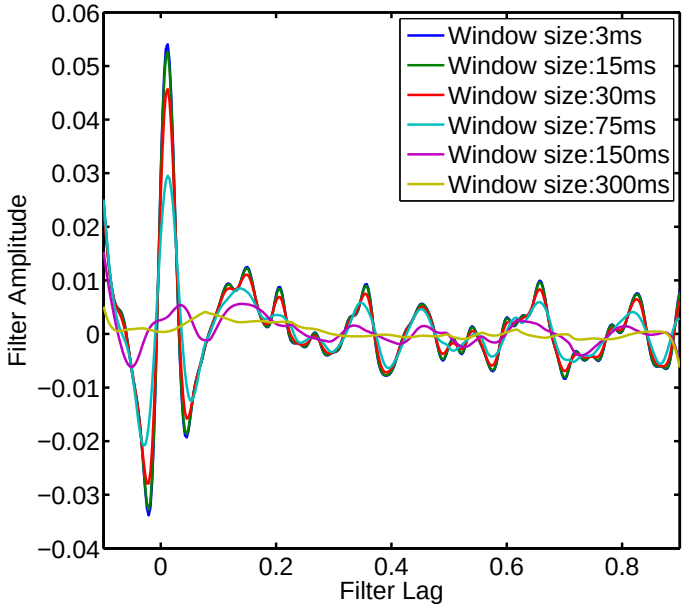
In the above analysis, we have considered how a boxcar average over the stimulus immediately preceding the current time affects gain. Now, we want to find some optimal way of averaging the past stimulus history to predict the instantaneous gain: by doing so, we can then predict gain, and thus get a better predictor of the actual firing rate.

In effect, we can calculate a new filter, K_g such that

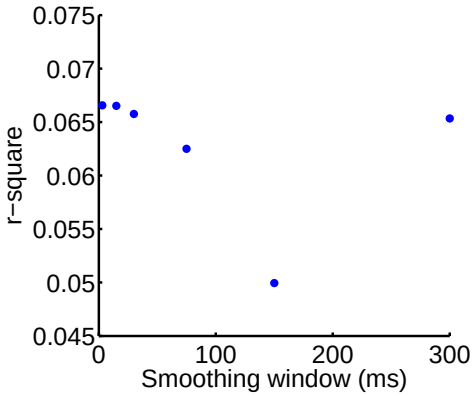
$$K_g = \hat{C} \setminus (s' * g)$$

where g is the smoothed gain and s is the stimulus.

For with various smoothing of the instantaneous gain, we find the best filter for each and plot it below:

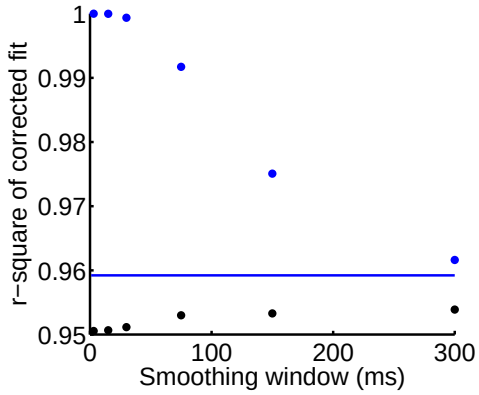


How well can we predict the smoothed gain vectors using these filters and the stimulus? The following plot shows the r-square between prediction of the smoothed gain and the smoothed gain for various smoothing windows.



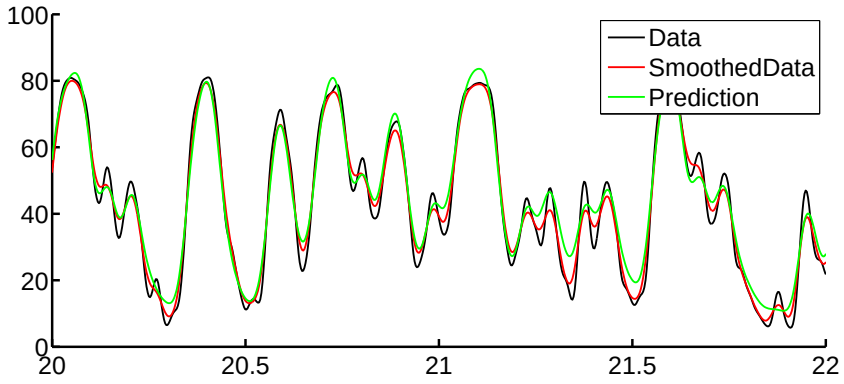
Well, that sucks. Maybe even this horrible estimation of the gain can improve the linear prediction of

the firing rate in some way? Here, I plot the r-square of the gain-corrected linear prediction (corrected by the prediction of the gain), as a function of gain smoothing (in blue). Also plotted is the r-square of the gain-corrected linear prediction, corrected now with actual gain (in black).



Re-evaluating filter performance after filtering ORN responses

The ORN firing rate traces are quite noisy, and it is possible that this is screwing up our estimation of the gain. It is also possible that the linear filter is doing a really good job and can't really be improved, and all the gain residuals are just noise. To check for these possibilities, I will filter the ORN response with a small sliding window, and then recalculate everything.



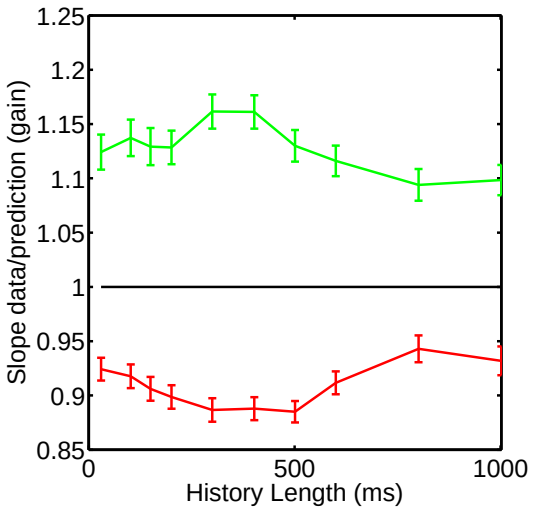
The r-square of the smoothed ORN response to the actual data is

```
0.9846
```

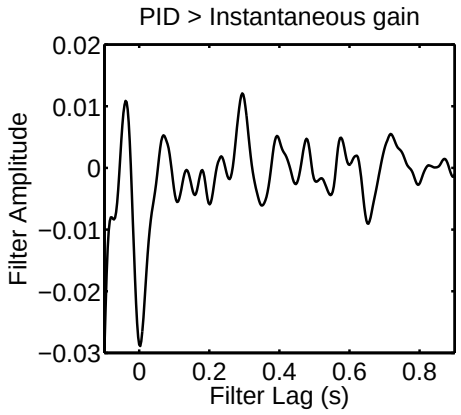
and the r-square of the smoothed ORN response to the predicted smoothed data is

```
0.9723
```

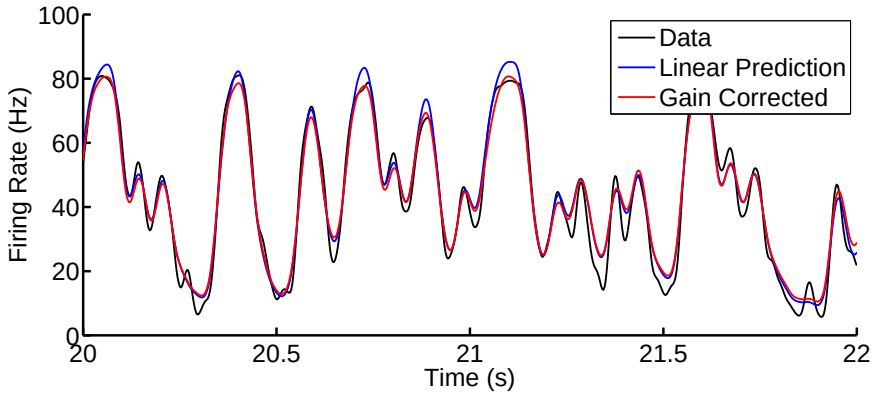
The smoothing operation should not interfere with the gain modulation we observe. Here, we repeat the gain analysis, but this time with the smoothed data and the smoothed prediction.



Once again, we can estimate the instantaneous gain and construct a filter from the PID to the instantaneous gain. This filter looks like



This filter can be used to correct the simple linear prediction from the original data. Now, we can compare the gain-corrected prediction to the *original unfiltered firing rate* and see if is better than the simple linear prediction.



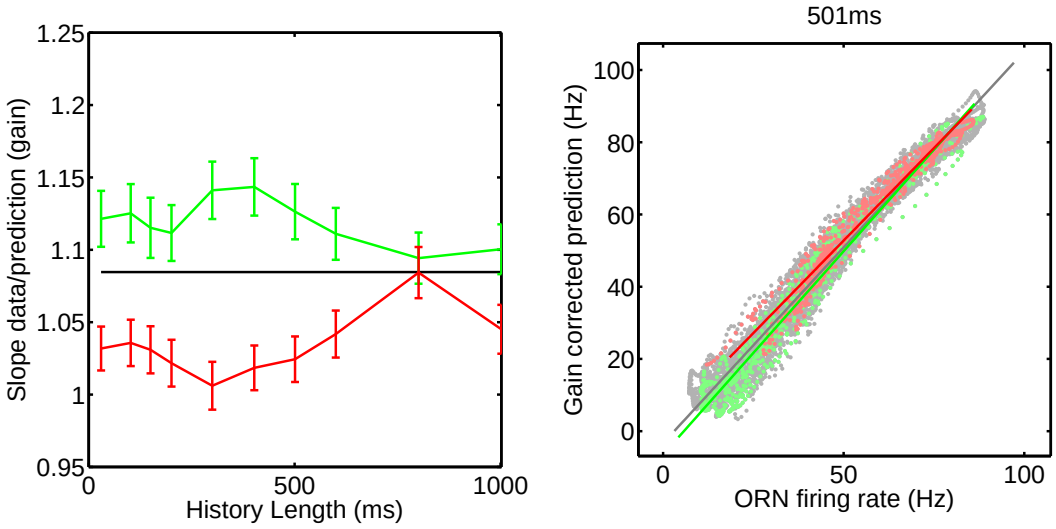
The r-square of the gain-corrected prediction to the data is

```
0.9616
```

cf. the r-square of the simple linear prediction to the data which is

```
0.9592
```

Does this gain correction make the gains to low and high stimuli the same? In the following plot, we plot the best-fit lines of corrected prediction to data for different history lengths, and also show an example scatter plot for the gain corrected prediction.



A note on elastic net regularisation

Carlotta says in her paper that she uses elastic net regularisation (using some unknown, third party code) to minimise the L-1 and L-2 norms of the covariance matrix. However, her code shows that she uses simply L-2 minimisation as discussed previously. I talked to her, and she said that she arbitrarily chose parameters for regularisation anyway, and it is possible that she was using simple L-2 regularisation, as this is a subcase of the elastic net regularisation.

Next Steps

1. Check if Weber's Law is being followed using Carlotta's step responses data
2. Run analysis on all data files.
3. Fit DA model to data with tau set to zero

Docs

This document was generated by MATLAB's *publish* function. All files needed to generate this document are on a git repository. To clone onto your machine, use:

```
git clone https://srinivasgs@bitbucket.com/srinivasgs/da.git
```

You will also need a bunch of functions that this depends on, which are also on git. Use

```
git clone https://srinivasgs@bitbucket.com/srinivasgs/core.git
```

to get these.

Once you have everything, run these commands to generate this document:

```
options = struct('showCode',false,'format','latex','imageFormat',...
'pdf','figureSnapMethod','print','stylesheet','srinivas_latex.xsl');

publish2('Analysis_January.m',options);
```